Advanced Teaching of Geometry with Interactive Tools (Innovative Teaching of Mathematics with Geometric Algebra)

AUTHOR(S):

Brehm, Enno; Kortenkamp, Ulrich

数理解析研究所講究録 1378 巻 2004 年 105-120

Advanced Teaching of Geometry with Interactive Tools

DFG Research Center Mathematics for Key Technologies
Technische Universität Berlin, Institut für Mathematik
Enno Brehm, Ulrich Kortenkamp

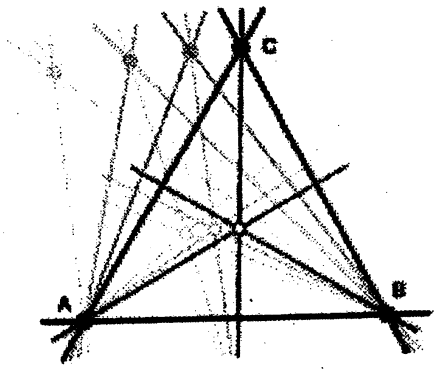
Abstract

In this article we will give a brief introduction into new ways of teaching that are possible with interactive geometry software, in particular with Cinderella, which has just become available in Japan.

1 Introducing Cinderella to Japan

Before we describe some new teaching scenarios, we will summarize the basic facts about the geometry software *Cinderella*.

1.1 Interactive and Dynamic Geometry Software



What is *Interactive and Dynamic Geometry*? With this term we refer to software that allows us to interactively do elementary geometry using electronic counterparts of ruler and compass in a dynamic fashion. That is, even after a construction is finished we can still move objects around and the construction will adapt itself to the changed positions. So there are not only geometric objects, but also *relationships* between them that are maintained all the time. The basic geometric objects include points, lines, circles, conic sections and more. One can imagine a DGS (short for *Dynamic Geometry Software*) as a geometry calculator. Cinderella [11] is a DGS written completely in Java, and thus available wherever there is a Java virtual machine. This makes it very suited for classroom use. You can find more information about Cinderella at `http://www.cinderella.de`.

One of the main features which sets apart Cinderella from many other geometry packages is its mathematical robustness: every construction behaves as "natural" as possible, and inconsistencies and jumping elements due to singularities are avoided whereever possible by a solid mathematical foundation. Moreover, it can also serve as an authors' tool to design web pages with interactive constructions or even complete geometry exercises.

Cinderella is in development for over 10 years by now, and the first commercial version is available since 1998. We refer to the manual of Cinderella [11] for a detailed overview about the origins and history of the software.

We also want to remark that we are still working on a next version of Cinderella. You can find more information about it in [6].

1.2 Cinderella in Japanese

Since autumn '03 a there is a japanese version of Cinderella available from Springer, Tokyo [13]. You should also refer to `http://cdyjapan.hp.infoseek.co.jp` for more information. Therefor, it is possible to use interactive geometry with Cinderella as described here not only in academic institutions, but also in japanese K-12 education. Especially in this context it is import to have localized versions of educational software since software in a foreign language may cause problems and impose an additional burden when starting to use computers in education and it is great for us that the japanese version of Cinderella may do pioneer work here. We see this as a form of cultural exchange because a lot of what makes Cinderella what it is now would not have been possible without technology from Japan, for example Cinderella running on handheld computers (see section 3.3).

The translation was done by Kazushi Ahara, and we are grateful to all our Japanese beta testers, in particular Ryosuke Nagaoka and Eckhard Hitzer.

2 Interactive Geometry Software in Teaching

Interactive Geometry Software has been shown to be very suitable for use in classrooms and teaching for many years now, and it is not possible to list all the relevant publications here. Geometry demonstrations gain very much from dynamics: this is for example easily seen in a simple construction showing that the bisecting lines in a triangle meet in a single point. If you are able to move the vertices of the triangle while the whole construction follows every movement, is is easy to demonstrate and experience that this assertion is always true.

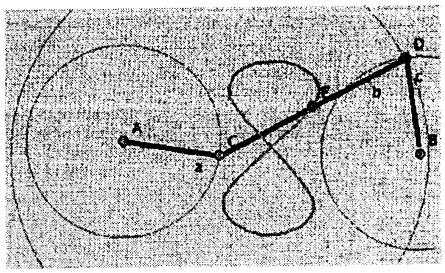
2.1 Experimental Mathematics

Usually, geometric constructions done with pen and paper require a large amount of precision and can be a lot of work, especially for complex constructions. Obviously, a major advantage of geometry software is that everything is still editable, mistakes can be undone, and the elements are not fixed to their places. This allows experimenting and even invites to do so, rendering possible constructions one probably would not even try on paper because they are too much work or too error-prone due to inaccuracy. So the range of feasible constructions and activities is greatly expanded.

But this still is a rather static advantage. The real potential lies in the dynamic power that a DGS provides. For example, if you do a construction and notice that three points lie on a line, you might ask yourself whether this happened accidentally or not. With pen and paper you would have to construct a different example of the same construction to check solely *one* more case. In constrast, using a software you can try to move around free elements of the construction to check what happens to the dependent points in different configurations. Of course, this kind

of experimenting is not a replacement for rigid formal proofs, but still it might give new insights and inspirations for research.

Nevertheless, this is a way to do mathematics which was not possible before the existence of computers, because what one actually does by continuously moving elements in a construction is to quickly check a very big number of examples.[1]



Locus of point E when C is moved on the circle

Another example where experimenting with dynamic geometric software can greatly expand your understanding are loci, i.e., to track how one point moves in dependency of other moving points. With Cinderella for example you can actually see which road a point would take while you move around objects that the point depends on.

This kind of mathematics might seem too much "play" to some, however, especially in the context of classroom use this aspect should not be neglected, because it makes mathematics which has a reputation of being very abstract and dry a little more tangible to many people, because you can actually grasp and manipulate objects and "watch" their behavior. This stands in the traditional spirit of the word *experimentation* which always included touching and manipulating things. Of course this "playful math" must always be correlated with traditional "hard" mathematics.

You find more examples and a deeper discussion of this in [7].

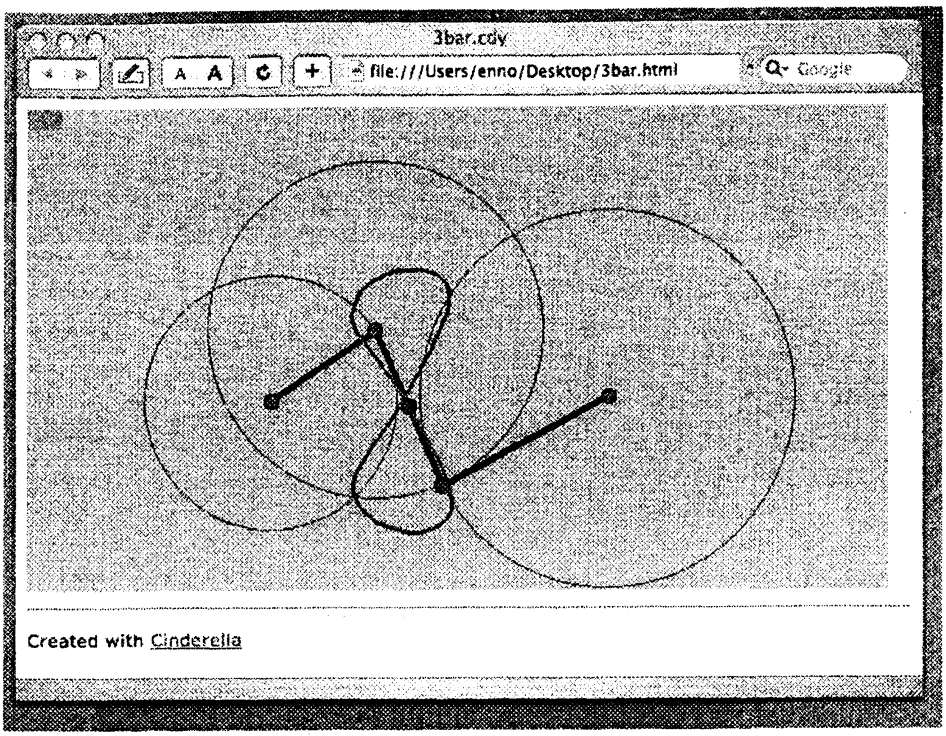
2.2 Mathematics on the Web

A good way of teaching can be to make students learn together and from each other. Again, the use of geometry software opens completely new possibilities. One contribution is that students can easily share their work, for example by sending constructions via e-mail or they can create images of their constructions and publish them on the web. However, images do not come up to the dynamic aspect as they are static by nature. Especially this is an inferior way to publish animations which can also be created with a DGS.

There are multiple ways to overcome this: one is to use animations instead of pictures which surely is an improvement over static images, but still this is not quite satisfactory. Fortunately there are different ways to present dynamic and even interactive content on the web. The most powerful of them is the use of Java applets, which actually allows to run programs within a web browser window.

Here the use of Java for a geometry software really pays off, because this way it is easily possible to bring interactive, dynamic geometry to the web: Cinderella allows to publish any construction on the WWW by automatically creating Java applets which can be put on web pages. Visitors of the page can not only see the construction but they even can interact with it in the same way as the user of Cinderella does.

[1] One can even show that this kind of checking many examples – contradicting mathematical intuition – can constitute a proof. See [3].



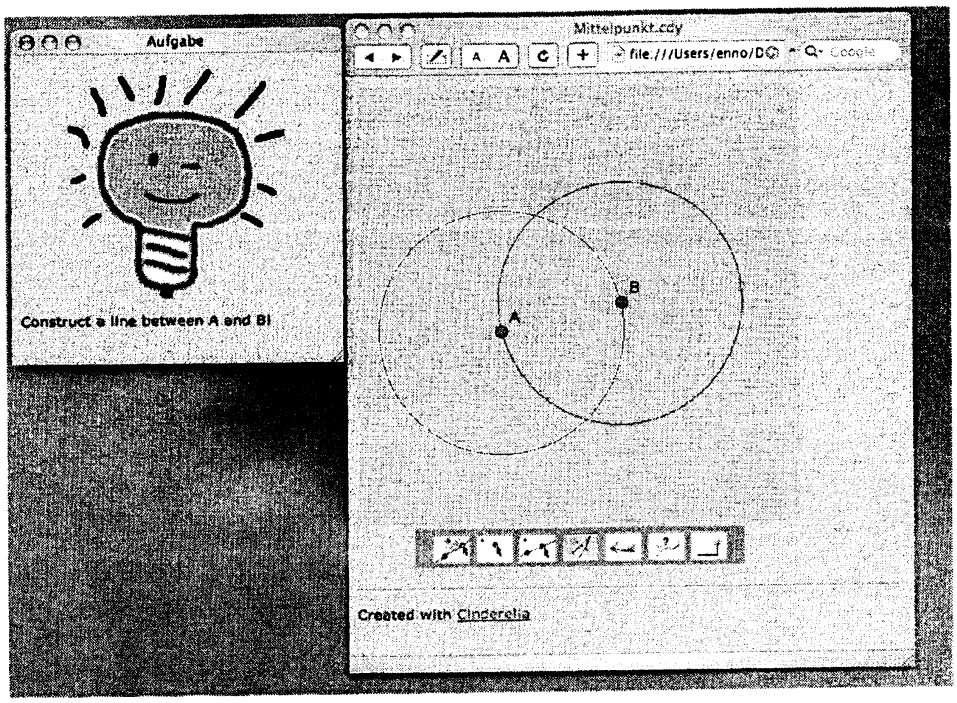
A construction as a Java Applet.

2.3 Interactive Exercises

Interactive illustrations are easy to do with these applets. This alone makes it possible to create, say, interactive electronic textbooks, which really offer added value over their printed counterparts[2]. But with Cinderella one can do even more: it is possible to create interactive *self-checking exercises* (which also run as an applet inside a web browser).

In such an exercise the student is presented with a geometric problem and some elements (points, lines) with which to start, and finally some tools out of the toolbox of Cinderella which he may use to solve the problem. The system additionally offers feedback and tutoring: whenever the student is stuck, he may ask for help and can be given a hint on how to continue. It is also possible to get instant positive feedback when he is on his way to the solution, and, of course, when he has completed the exercise successfully.

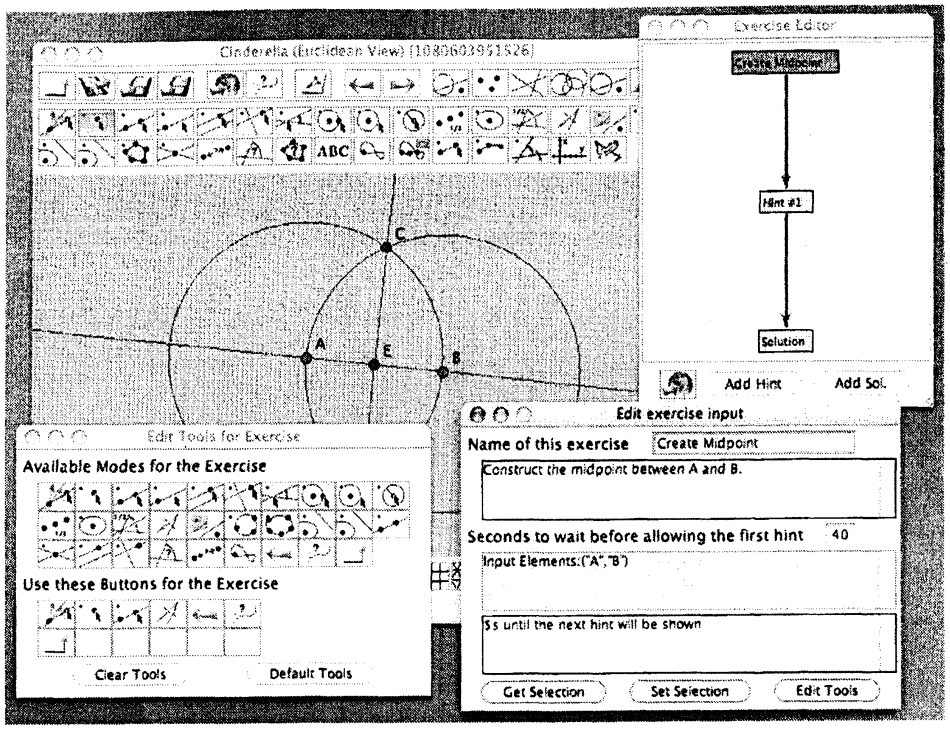
[2]And this is what computer use in teaching always has to be measured against, because it should never be an end in itself



An interactive exercise with a hint.

Naturally, all this has to be prepared carefully by a teacher who first has to do the correct construction herself and then turn it into an interactive exercise: she has to select the elements the student is supposed to start with and which tools will be allowed. For example: she wants to create an exercise in which the goal is to create the middle point between two points. In this case the starting point are simply two points. The conventional tools would be ruler and compass – so, in Cinderella one would add the tools for creating points and lines, the compass tool, for moving elements and probably the undo tool.

The tutoring elements have to be prepared by providing feedback text and selecting the elements after whose construction the respective texts will be shown. Similarly, hints have to be prepared, and it is possible to set time limits for these hints. This is to avoid that a student immediately asks for hints without even trying to solve the problem by himself.



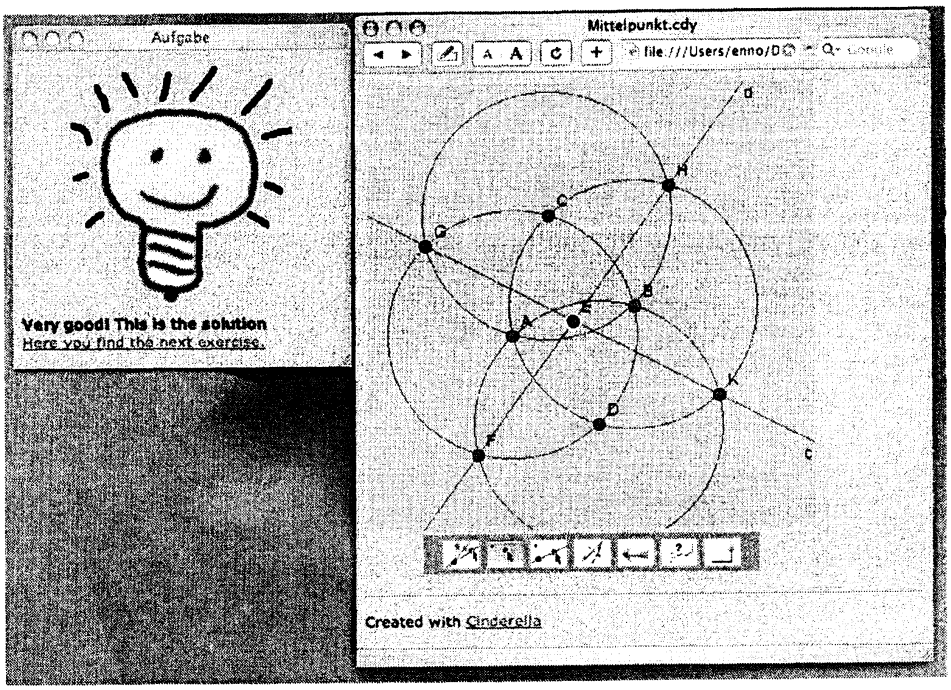
Exercise authoring component.

Finally the teacher has to specify the goal of the exercise by selecting the element that in the end has to be constructed by the student. Of course this solution is stored in the exercise so it seems to be easy to check whether a student solved the problem correctly, but in this context there a pitfalls to be avoided: on the one hand one has to make sure that the user did not just "guess" a solution (e.g., in the example just put a new point somewhere in the middle between the starting points).

On the other hand there might be multiple correct solutions for an exercise and although it is possible for the teacher to specify multiple solutions to the exercise, the system should be able to recognize and accept a correct solution, even if it was not one the teacher had in mind.

Cinderella does this by not comparing concrete positions of objects concrete steps to the solution, but by comparing the finally constructed elements: it proves internally whether the solution provided by the teacher and the solution of the student are equivalent even if they were constructed in a different fashion.

This gives the student the freedom to find his own solution. Additionally the fact that the system is dynamic helps the student in another aspect: after constructing elements he can always move elements around and convince himself that the constructed objects really have the properties he expects them to have, which adds to his self-confidence.



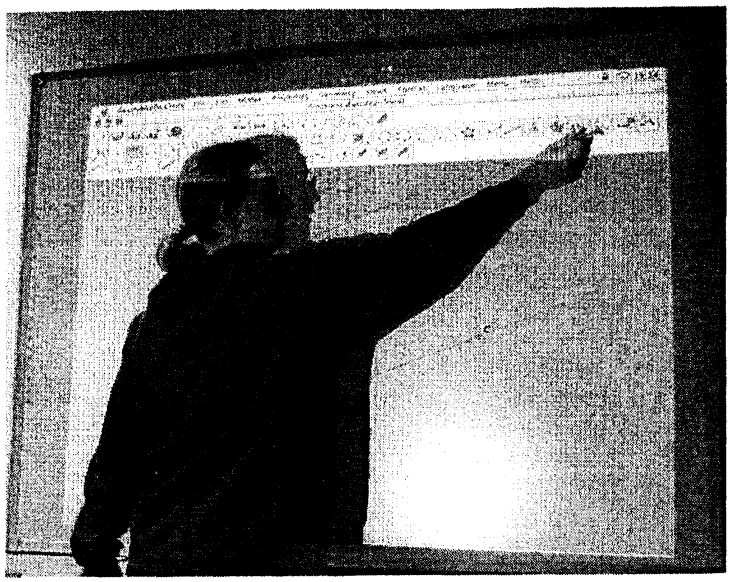
Non-standard solution to midpoint exercise.

3 Advanced Scenarios

Most of what we described so far is already present for several years now, see [5]. But DGS does not end at the screen of desktop computers. On the contrary: the real didactic power unfolds when used on other devices, ranging from large ones, like interactive whiteboards to very small ones like Personal Digital Assistants (PDA). Both of these have surprisingly much in common, user-interface-wise. We briefly sketch how this technology can be used to open up new ways of teaching mathematics and geometry, and have a look at how this can change how students or students and teachers can work and learn together. We refer to [9] and [4] for more information.

3.1 Interactive Whiteboards

The best device to do geometry for a large audience still is the blackboard (or whiteboard). In classroom or conference presentations nothing beats the direct interaction as shown there.



Interactive Whiteboard.

Currently there are different approaches to integrate whiteboards with computers. Among them is a very appropriate one for our purposes, that uses a specially equipped board as a large computer display (by using a projector) and a special pen that interacts with the board which functions as a mouse ore, more generally, as an input device. The whole setup is comparable to a mixture between a big touchscreen and a giant graphics tablet.

Using DGS on such a device can really be a new teaching experience: while preserving the advantages of the large presentation and drawing area, it still offers the modern assistance of a dynamic geometry software.

Since it is actually a computer screen projected on a whiteboard, you can use arbitrary applications on it. Anyone who can use a standard desktop computer will be able to use software on the interactive whiteboard. However, this advantage is at the same time a disadvantage since these standard user interfaces are not intended for such a large area. They are meant for being used with a mouse, where your hand does not have to move too much to make the mouse pointer travel across the screen. On the whiteboard, where the pen plays the role of the mouse, this kind of interface can be rather cumbersome, e.g., to access a menubar you have to physically stretch to the top of the board.

This is a widely known problem and there are different approaches to solving it. One way is to think of new user interface elements that do not require large movements, e.g., circular menus that appear right where the pen is[3], or a system to recognize gestures of the pen, like a quick drag of the pen from right to left to undo an action or a large zigzag move across the whiteboard to clear the current construction and start a new one[4].

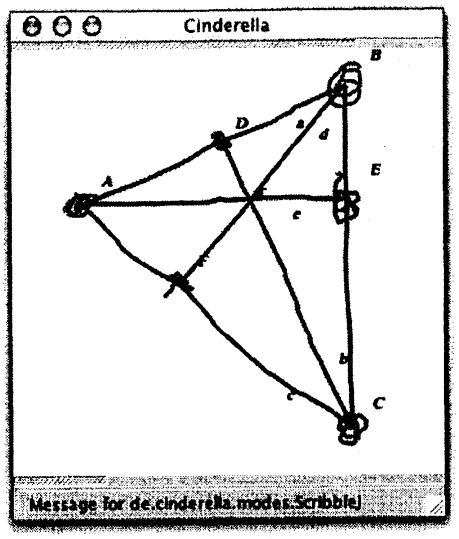
3.2 Sketching

While gestures are a rather generic approach there are also ways of improving the user interface which are a lot more specific to the problem domain of geometry: the classic way to create

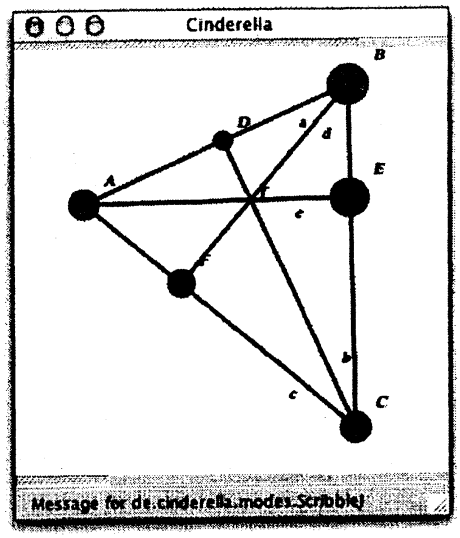
[3]We have actually implemented such a menu scheme which is usually known as flowmenus[1], see [9]

[4]This gestures was chosen because it resembles a "cleaning-a-blackboard" move, so it would be very intuitive

geometry is to simply *draw* it. So we tried to recreate the swiftness of hand drawing while maintaining the advantages a DGS has to offer.



Sketch drawn by hand...



...Automatic recognition of sketch

Sketch recognition is a useful technique for all pen-based devices, like whiteboards, graphics tablets, tablet PCs and handheld computer with pen

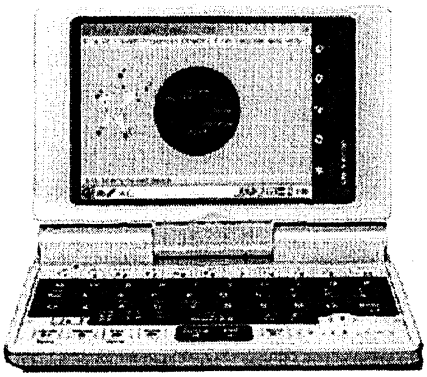
This is done by interpreting a hand drawn sketch and automatically transforming it into a construction, whereas the hand drawn objects can either be kept or be replaced by their "perfect" counterparts. The goal was to create an interface where as much as possible could be done without the use of special tools or modes, respectively. Consequently we had to develop different ways to let the DGS know about what the sketch actually was supposed to mean[5]. This was achieved by letting the user annotate the drawing in a way as natural as possible . For example, to indicate orthogonality of two lines, the user would draw a small orthogonality-mark at their intersection – exactly the same she would do one a real sketch-pad.

3.3 A PDA version of Cinderella

Even closer to such a real sketch-pad comes the smallest devices we see DGS currently running on: personal data assistants (PDA) or handhelds. While being a lot smaller than an interactive whiteboard, the user interface difficulties are similar, albeit for different reasons: these pen-based handhelds usually have a 3x4 inch touch-sensitive screen (about 240x320 pixel), so real screen estate is very limited.

Therefor using a standard desktop application on such a screen does not really make sense, since the standard user interface elements are too small, and thus hardly usable with a pen. Increasing their relative size will leave even less space for the drawing, so this is not an otion. Fortunately, most techniques to improve usability of the interactive whiteboard can be transferred to this device, especially gestures work very well here. Usability in general is still an

[5]This information is usually provided by the user by selecting the right prior to the construction of elements



SHARP(tm) Zaurus SLC700 running Cinderella

issue on these devices and it seems that each application has to find its specific way to optimally use the limited input capabilities. For geometry, sketching looks like a very promising approach.

3.4 Collaboration

A traditional way of student collaboration is offered by small workgroups: a few students sitting together, working on a problem. Here again DGS show their strength as they can intensively work together by sharing a single computer: they can communicate their ideas and try things out immediately. Pen and paper would take a lot more time, especially if accuracy is required.

However, more innovative ways of using tools can be achieved by combining the technologies mentioned above: if there is a DGS on a desktop computer, on the whiteboard and on the handheld, they should be able to work together: they could effortlessly connect to each other and share one construction, such that every change a user makes on one device could be seen immediately on all other ones[6].

As another example imagine a teacher at an interactive whiteboard, demonstrating and explaining a problem, that the students automatically have on their own device, be it a PC or a handheld, and can work on it isolated or in small groups. Later for presentation the solutions found by the students could immediately be made visible again on the whiteboard for all to see without the necessity of transferring any files[7]. This works together nicely with wireless connection technologies which are getting more and more ubiquitous.

The same technology can allow students to help each other in their homework, even if they are not at the same location: they could connect their geometry programs on their computers at home to share a single construction via the internet, and work together and answer each others questions. Combined with video telephony or similar software this could really give completely new ways of working and learning together. This idea stretches even further: one can easily imagine remote teaching scenarios, in which a teacher gives a lecture on an interactive whiteboard which in turn is linked to another one somewhere else with people following the lecture

[6]This is already working and will be available to users in version 2.0 of Cinderella

[7]Again, care has to be taken in such scenarios not to overuse technology. We do not want to avoid students coming to the front of the class and explaining things directly at the whiteboard.

and the possibility to "come to the board" and interact with the teacher. (A system supporting this kind of remote teaching already is, for example, e-Chalk.[14]. Cinderella already has been shown to integrate well in this system).

At this point we would like to point out that these experimental features still have to be evaluated in the classroom, and we cannot be sure that they will really have a didactical impact.

4 Outlook and Conclusion

So far we have seen how the teaching of mathematics may be advanced by the use of a DGS in general or Cinderella in particular. We now give a perspective of some aspects to come which might not be immediately available to teaching but still are interesting to keep in mind.

4.1 Visualization of Algorithms with DGS

Discrete mathematics is becoming more and more important in schools. This has a good reason: it is a mathematical branch, albeit a relatively young one, where problems usually are often easily formulated and easily grasped (but not necessarily easy to solve). It has many real-world applications, and last but not least, it motivates students by giving them something to puzzle with. Nevertheless, it is possible to bring most basic mathematical concepts across using Discrete Mathematics. Moreover, algorithms play an important role in Discrete Mathematics. There exist many geometric algorithms and graph algorithms, which often can be understood geometrically. So there is a strong connection between algorithms and geometry.

When learning about algorithms, one crucial point is to understand how it actually works and what the basic idea of the algorithm is. Unfortunately, algorithms often are presented in form of pseudocode[8]. While being precise, this form of notation tends to hide the central idea of an algorithm. This is because it is closer to the way a computer would actually execute it then to the idea the author had when inventing it. But exactly this idea is the most important thing to learn – once it is understood, it may be very easy to recreate the complete algorithm. A simple example for this is the well-known Quicksort [2] algorithm which is based on an ingenious but simple idea[9], which is much harder to understand when reading its pseudocode.

The best way of course to understand an algorithm is to have its idea explained by a teacher or someone else who already understood it. But sometimes it is necessary for a student to work alone and then it can be helpful to be able to "simulate" an algorithm. Or a teacher might want to explain the algorithm supplemented by a simulation. Again, doing this with pen and paper is tedious and neglects their dynamic nature, or is simply not practical (e.g., in a classroom), so the use of a computer to simulate and visualize the workings of an algorithm and its data structures is self-evident.

One way to follow an algorithm is to step through its code, seeing it work line by line with the help of a debugger. You can pause at each step and see which variable holds which value.

[8] Pseudocode is a semiformal notation, which borrows elements from mathematical notation and imperative programming languages

[9] Basically the idea is the following: to sort an array of numbers, pick an arbitrary element, move every number that is larger to the right, and all others to the left. Then apply the same principle recursively to the two subarrays, that you just obtained.

```
partition(a, left, right, pivotIndex)
    pivotValue := a[pivotIndex]
    swap(a[pivotIndex], a[right]) // Move pivot to end
    storeIndex := left
    for i = left to right-1
        if a[i] <= pivotValue
            swap(a[storeIndex], a[i])
            storeIndex := storeIndex + 1
    swap(a[right], a[storeIndex]) // Move pivot to its final place
    return storeIndex

quicksort(a, left, right)
    if right-left+1 > 1
        select a pivot value a[pivotIndex]
        pivotNewIndex := partition(a, left, right, pivotIndex)
        quicksort(a, left, pivotNewIndex-1)
        quicksort(a, pivotNewIndex+1, right)
```

Pseudocode for Quicktime Algorithm (from [16])

Unfortunately, this is often too much and too fine information – basically it is the wrong level of abstraction because it shows more of what the computer sees of the algorithm.

What in fact could be helpful is customized visualization, where the less important information is hidden and a the spectator gets focused on the important parts. Creating such a custom visualization of an algorithm culminates in even more work than just simulating it by hand[10]. So what is needed is a way to visualize an algorithm as reasonable as possible with as little effort as possible.

Obviously this is a very generic problem, but if one focuses on geometric and graph algorithms a DGS respectively Cinderella with its rich support for drawing and manipulating geometric objects forms a solid base to create visualizations of such algorithms. One approach to do that is to augment the algorithms code, like in this incomplete example, which is taken from an algorithm to calculate a minimum spanning tree (MST) in a weighted graph[11]

```
private void visitNeighbors(Vertex vertex) {
    EdgeVector neighbors = g.outgoing(vertex);
    for (int i = 0; i < neighbors.number(); i++) {
        Edge nextStep = neighbors.get(i);
        if (nextStep.getColor().equals(NOT_VISITED))
            visit(nextStep);
    }
}
```

[10] And if you were actually able to create it, you will not need it anymore

[11] The algorithm selects a set of edge that connect all vertices without forming a cycle, such that among all possible choices the sum of the weights of the selected edges is minimal.

```
private void visit(Edge edge) {
    flash(edge, VISITING, 500);
    if (edge.to().getColor().equals(NOT_VISITED)) {
        edge.setColor(EDGE_IN);
        edge.to().setColor(VISITED);
    } else {
        edge.setColor(EDGE_OUT);
    }
    stepDone();
    visitNeighbors(edge.to());
}
```

In this code snippet the author had to insert code concerned with simulation and visualization respectively. (e.g., `flash(...)`, `edge.setColor(...)`) in between the code representing the logic of the actual algorithm. This is the crucial point when visualizing an algorithm. What we think is necessary is a tool which aids in this process, such that in the ideal case, the algorithms code is analyzed and presented in a way, so that the user/teacher can specify in an abstract fashion what and when to visualize, without immediate need to edit the code to obtain a meaningful visualization. A prototype for such a system has to be build and evaluated. A similar approach of semiautomatic animation is shown in [15].

4.2 School-PDA

In section 3.3 we already discussed the advantages of handheld devices equipped with Cinderella and mentioned their use in possible collaboration schemes 3.4. But the devices currently available are rather tailored for business needs. To really exploit their potential in classroom use and for the student in general, a PDA would be needed that comes with software preinstalled especially chosen for school use, which could include, besides a DGS/Cinderella, a computer algebra package like e.g., MuPAD [10] and packages for other subjects, like vocabulary trainers etc. The japanese company SHARP already showed interest in cooperating to design such a PDA, which could be of great use in teaching if reasonably used. We seriously hope that this project can be realized.

4.3 Natural language input

Geometric constructions (like all mathematics) can be expressed in a purely formal, completely unambiguous language. Still, this is not how human beings talk about geometric circumstances nor is it the way that is appropriate for them. Being able to describe something concisely and in an unambiguous fashion is a fundamental skill students have to learn. While it cannot be expected in the near future that computers will be able to analyze (or even come close to "understand") natural human language (even if it is input textually), a specific problem domain with a constrained vocabulary like geometry could be a starting point.

An experimental parser has been implemented [8] that accepts descriptions of geometric constructions written in natural language and tries to reconstruct the corresponding objects, so students could control their own descriptions by checking the construction that gets built up. This project was as ambitious as it sounds, and it became obvious rather quickly that realising

it completely was out of the question. Nevertheless the parser worked astonishingly well for some descriptions – unfortunately it would also accept many sentences that were nonsense, a problem which made it unsuitable for classroom use where one would rather enforce correct language. Besides it seemed very hard to transfer concepts to other input languages (the parser was written for german).

What has been learned from this once more, is that a very long way is still to go to make computers accept the most natural forms of human communication[12], but from the achieved results, one can get a short impression of what could be, even if it was only for a limited subset of geometry.

4.4 Scripting & Macros

While graphical user interfaces and the mouse provide a very comfortable way to enter and manipulate geometric objects, they have their boundaries, too. Especially when it comes to repetitive tasks or constructions involving a large number of similar objects a user might come to the point where she doesn't want to do the same over and over again. Or she might have a short and concise description of how some large group of objects could be created, like e.g., a square grid of 20 by 20 points with different colors. The next version of Cinderella will contain scripting, such that creating the mentioned grid can simply be achieved by a script like the following:

```
for i in range(20):
        for j in range(20):
                p = createPoint(i,j)
                p.color = Color( i * 0.1, j*0.1, 0 )
```

Furthermore one will be able to save parts of a construction in form of such a script so it can be recalled or bound to a button and reapplied to other elements. This is achieved by embedding a widely used scripting language called "Jython"[13]. This alone might not seem worth mentioning, but it is important under an educational aspect, too: this scripting language is easy to learn and it is very readable[14], so it qualifies for being a first programming language. A start in programming could then be a "geometric" program like:

```
A = createPoint()
B = createPoint()
C1 = CircleWithCenterAndPoint(A,B)
C2 = CircleWithCenterAndPoint(B,A)
(P0, P1) = intersectCircles(C1, C2)
l = Line(P0, P1)
```

This would create the perpendicular bisector of the segment between to new points *A* and *B*. This does not really look like a program but more like a readable description of the construction, so "programming" in this case is *describing* – a key skill (see also 4.3)! By going one step further and introducing functions like

[12]Even disregarding voice recognition

[13]A dialect of the language "Python"

[14]In fact it is said to be pretty close to the pseudocode mentioned in 4.1

```
defun perpendicularBisector(A,B):
        C1 = CircleWithCenterAndPoint(A,B)
        C2 = CircleWithCenterAndPoint(B,A)
        (P0, P1) = intersectCircles(C1, C2)
        return Line(P0, P1)
```

students can learn to identify substructures and the principle of abstraction, both of which are fundamental concepts in mathematics. Another didactic advantage is that students can quickly be given programming tasks from a real problem domain and can actually use their results immediately to expand the capabilities of their geometry software. This can be a great motivation, for in many programming courses a student often either has to start by writing practically useless toy programs that write "Hello World" on the screen, or he has to fiddle around with GUI libraries which are too complicated to be understood by a novice programmer.

4.5 Conclusion

We have tried to show how the use of interactive tools or DGS in particular can really enrich the teaching of mathematics. Some of these uses are close to geometry, while some others might only be prototypes of general techniques (like interactive whiteboards) that we applied to mathematics and geometry. On the other hand we have also shown that a DGS is not necessarily constrained to doing pure geometry: since so many things may be understood oder modeled geometrically, it can be put to use in different contexts as a workhorse. And we are sure that we are just scratching at the surface of the potential of interactive tools like DGS for mathematics, and with the development of technology a lot more may be on its way.

We would like to acknowledge the support of the DFG research center Mathematics for Key Technologies Berlin, and the RIMS Kyoto, and we would like to thank the organizers of the conference for making all this possible.

References

[1] François Guimbretière, Terry Winograd. FlowMenu: Combining command, text entry and direct manipulation. UIST 2000

[2] Hoare, C. A. R. Partition: Algorithm 63, Quicksort: Algorithm 64, Find: Algorithm 65. Comm. ACM 4, 321-322, 1961

[3] Ulrich Kortenkamp. Foundations of Dynamic Geometry. Dissertation, ETH Zürich, 1999.

[4] Ulrich Kortenkamp, Dirk Materlik. Geometry Teaching in Wireless Classroom Environments using Java and J2ME. Accepted for publication in: *Science of Computer Programming, Special Issue on Practice and Experience with Java in Education*, Elsevier.

[5] Ulrich H. Kortenkamp and Jürgen Richter-Gebert. Geometry and education in the internet age. In *Proceedings of ED-MEDIA 98, Freiburg, Germany*. AACE, 1998. `http://www.cinderella.de/papers/geo-i.pdf.gz`.

[6] Ulrich Kortenkamp and Jürgen Richter-Gebert. Making the move: The next version of Cinderella. In Arjeh M. Cohen, Xiao-Shan Gao, and Nobuki Takayama, editors, *Proceedings of the First International Congress of Mathematical Software*. Singapore, World Scientific, 208-216 (2002). A slightly modified version appeared in the proceedings of CCCG 02.

[7] Ulrich Kortenkamp. Experimental Mathematics and Proofs – What is Secure Mathematical Knowledge? Submitted to : *Zentralblatt für Didaktik der Mathematik, Special Issue on Discrete Mathematics and the Role of Proof in the Classroom.*

[8] Dirk Materlik. Erkennung natürlicher Sprache für die interaktive Geometriesoftware Cinderella. Studienarbeit, Freie Universität Berlin, 2003.

[9] Dirk Materlik. Using Sketch Recognition to Enhance the Human-Computer Interface of Geometry Software. Diploma thesis, Freie Universität Berlin, 2003. `http://page.mi.fu-berlin.de/~materlik/DirkMaterlikThesis.pdf`.

[10] MuPAD. A modern, full-featured computer algebra system. `http://www.mupad.com`.

[11] Jürgen Richter-Gebert and Ulrich H. Kortenkamp. *The Interactive Geometry Software Cinderella*. Springer-Verlag, Heidelberg, 1999. `http://www.cinderella.de`.

[12] J. Richter-Gebert. "Mechanical theorem proving in projective geometry," Annals of Mathematics and Artificial Intelligence, 13, 1995, pp. 139-172.

[13] Jürgen Richter-Gebert and Ulrich H. Kortenkamp. シンデレラ日本語版. Springer-Verlag, Tokyo, 2003. Japanische Übersetzung. `http://cdyjapan.hp.infoseek.co.jp/`.

[14] Raúl Rojas, Lars Knipping; Ulrich Raffel; Gerald Friedland Elektronische Kreide: Eine Java-Multimedia-Tafel für den Präsenz- und Fernunterricht. Inform., Forsch. Entwickl. 16, No.3, 159-168 (2001). see also `http://www.e-chalk.de`.

[15] Alexander Schliep et. al. Gato - Graph Animation Toolbox. `http://www.zpr.uni-koeln.de/~gato/index.html`

[16] Wikipedia Entry. `http://en.wikipedia.org/wiki/Quicksort`.